Ausbildung Fachinformatiker Anwendungsentwicklung

Programmierübung

C–Programmierung

1 Aufgabe

Es ist ein Programm (`myfilter.c`) zu schreiben, das den Namen einer beliebigen Text–Datei/Pfad

1. als Parameter einliest,
2. diese Datei öffnet,
3. deren Inhalt einliest,
4. bestimmte Operationen ausführt und
5. das Ergebnis in die Standard–Ausgabe schreibt.

Der Datei/Pfadname kann entweder ohne Parameterkennzeichnung übergeben werden oder mit einer Parameterkennzeichnung (`myfilter -i <inputfile>` oder `myfilter <inputfile>`). Weitere optionale Parameter müssen von dem Programm eingelesen werden können. Die Parameter bestehen jeweils aus einem Buchstaben und sind durch ein vorangestelltes - gekennzeichnet. Diese Parameter bestimmen, welche von den u.g. Operationen ausgeführt werden sollen. Werden keine optionalen Parameter eingegeben, so soll eine `usage()`–Meldung ausgegeben werden (`./myfilter [-c|-w|-l] [-i] inputfile`). Folgende Operationen sollen durch das Programm durchgeführt werden können:

Parameter	Operation	Ausgabe
kein	mache garnichts	`usage()` ausgeben
-c	Zähle alle Zeichen	Zeichen=nnnn
-w	Zähle alle Wörter	Wörter=nnnn
-l	Zähle alle Zeilen	Zeilen=nnnn

Das Programm soll so konzipiert werden, dass später gegebenenfalls weitere optionale Parameter leicht hinzuzufügen sind. Außerdem soll das Programm als *Filter* arbeiten können, d.h. falls kein Datei oder Pfadname als Parameter angegeben wurde, soll das Programm seine Eingabe aus dem Kanal Standard–Eingabe beziehen. Das Programm soll seine Ausgabe immer in die Standard–Ausgabe schreiben. Einige mögliche Aufrufe könnten z.B. folgendermaßen aussehen:

```
bash$> ./myfilter       ./7zara10.txt
bash$> ./myfilter -l    ./7zara10.txt
bash$> ./myfilter -w    ./7zara10.txt
bash$> ./myfilter -l -w ./7zara10.txt
bash$> ./myfilter -c    ./7zara10.txt
bash$> ./myfilter -c -i ./7zara10.txt
bash$> ./myfilter
bash$> cat ./7zara10.txt | ./myfilter -l
```

Zum Programm soll eine etwa 3–seitige Dokumentation erstellt werden, in der die Arbeitsweise des Programms kurz skizziert wird.

2 Bemerkungen und Tips

Die Entwicklungsumgebung ist *MinGW*. Das Testen sollte in einer *bash*–Umgebung stattfinden. Entweder stellt *MinGW* diese zur Verfügung oder man öffnet eine *Gitbash*, in der man arbeiten kann.

Für das Einlesen der Daten ist die Funktionen `getc()` gegenüber der oftmals instabilen (und komplizierteren) `scanf()`-Funktion vorzuziehen. Zum Einlesen von Optionen und Argumenten gibt es unter Unix die Funktion `getopt()` .

Zur besseren Struktur und um ein besseres Testen zu ermöglichen, unterteilen Sie das gesamte Programm in mehrere Funktionalitäten und testen Sie jeweils nur die einzelnen Funktionen für sich:

- Einlesen und Verarbeitung der Parameter (`getopt()` verstehen!)
- Bestimmen der Ein– und Ausgabe–Kanäle (Kanäle und Umleitungen verstehen!)
- Unterfunktion zur Zählung der Zeichen, Worte und Zeilen (C–Programmierung und Zeiger)

Unter Umständen erzeugen Sie verschiedene Programme, in der nur eine bestimmte Funktion ausprogrammiert wird und fügen diese dann später zusammen.

Testen Sie oft und gründlich und berücksichtigen dabei möglichst viele Testfälle. Gehen Sie auf die von Ihnen durchgeführten Tests auch kurz in der Dokumentation ein.

3 Abgabe

Die lauffähigen Programme und der Quelltext sind mit einer etwa 3–seitigen Dokumentation abzugeben (ins Repository: C⇒ Eingang⇒ <Name>, Dokumentation in LaTeX oder `ASCII`).

4 Zeitrahmen

Der Zeitrahmen dieser Aufgabe ist mit etwa 1–2 Wochen angesetzt.